Fig. 1

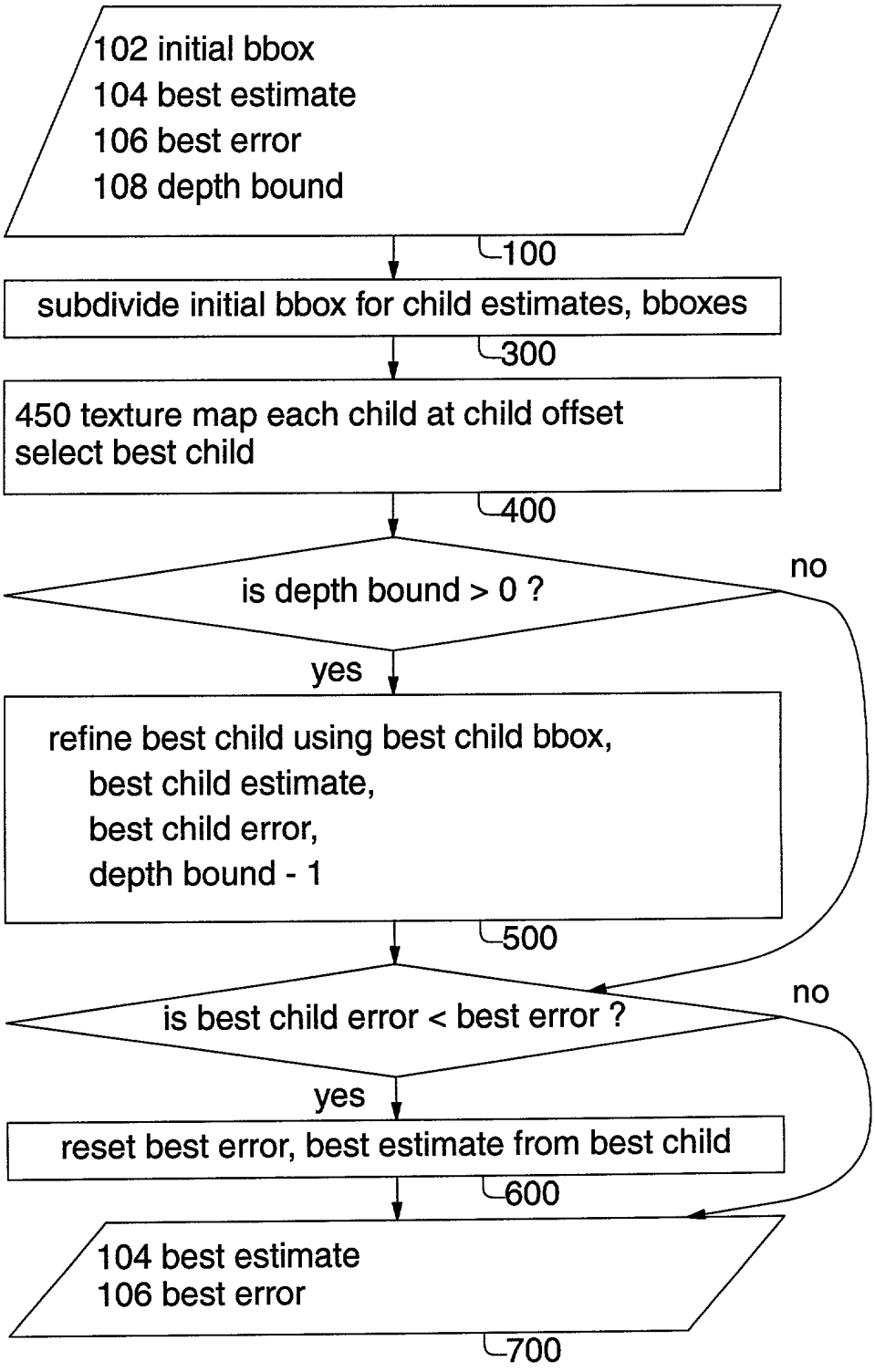

102 initial bbox
104 best estimate
106 best error
108 depth bound
100
subdivide initial bbox for child estimates, bboxes
300
450 texture map each child at child offset
select best child
400
is depth bound > 0 ?
no
yes
refine best child using best child bbox,
best child estimate,
best child error,
depth bound - 1
500
is best child error < best error ?
no
yes
reset best error, best estimate from best child
600
104 best estimate
106 best error
700

Fig. 2

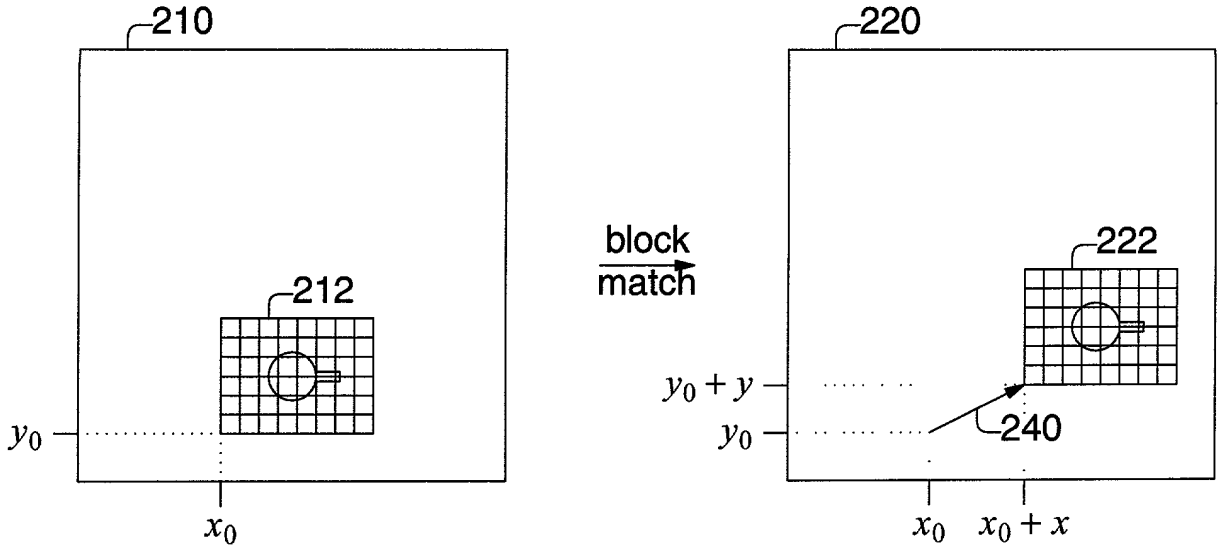

210
212
y_0
x_0
220
block
match
222
$y_0 + y$
y_0
240
x_0
$x_0 + x$

Fig. 3

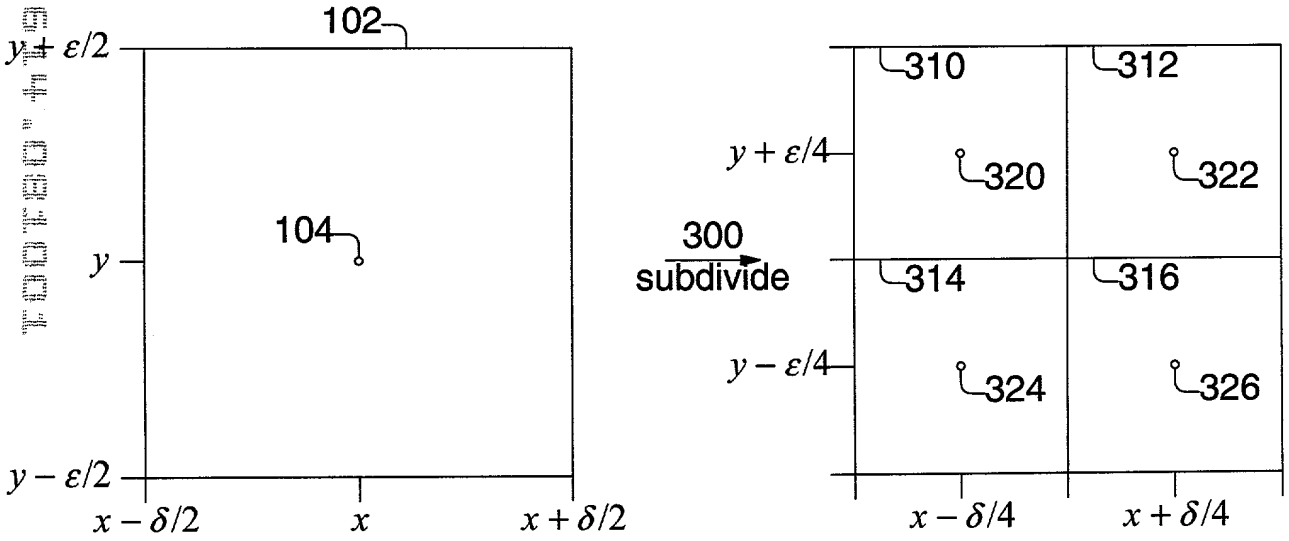

102
$y + \varepsilon/2$
104
y
$y - \varepsilon/2$
$x - \delta/2$
x
$x + \delta/2$
300
subdivide
310
312
$y + \varepsilon/4$
320
322
314
316
$y - \varepsilon/4$
324
326
$x - \delta/4$
$x + \delta/4$

Fig. 4

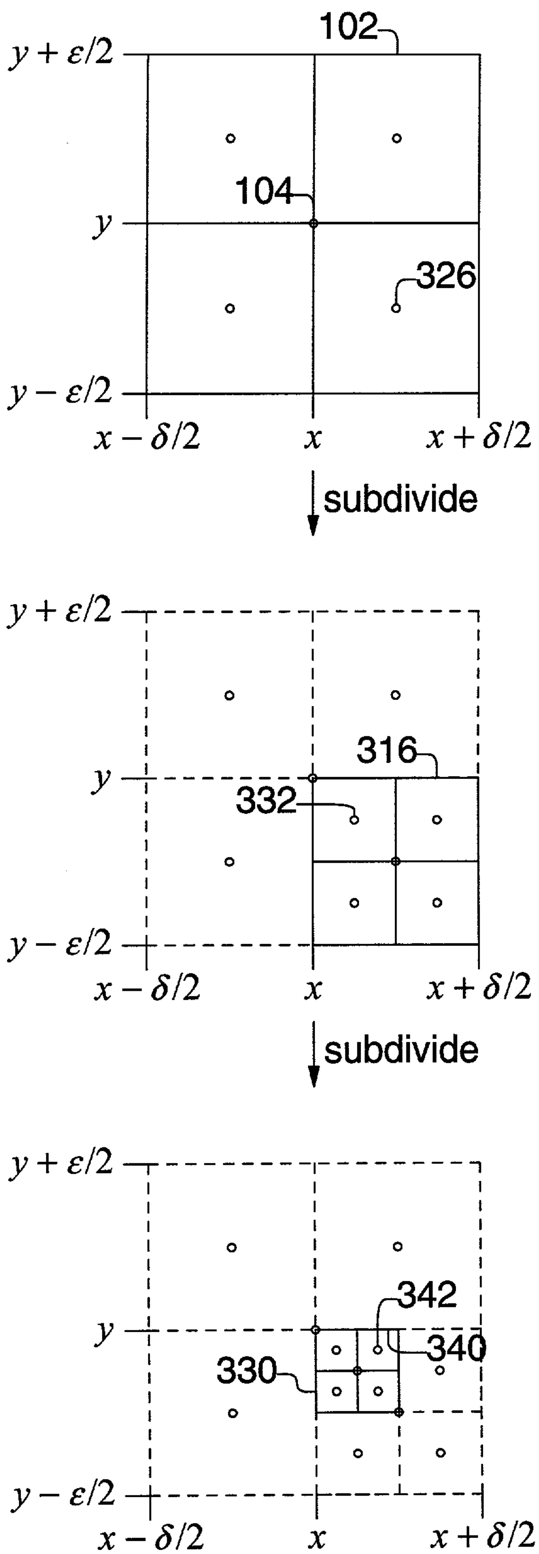
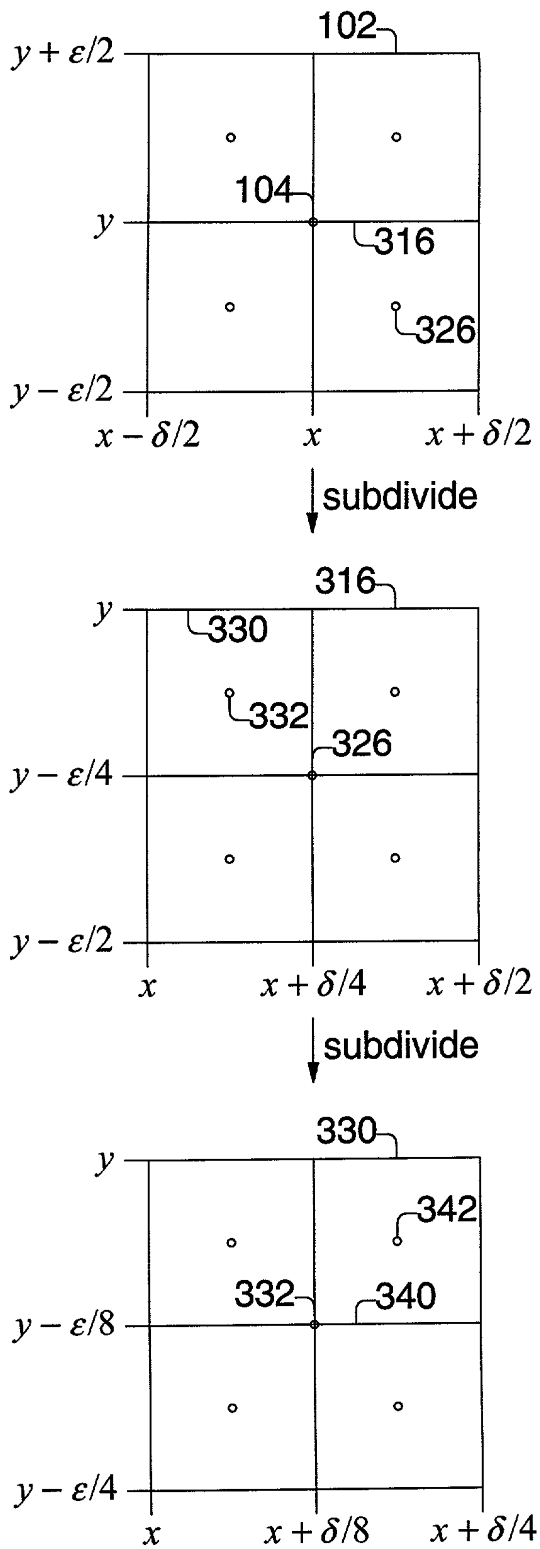

102
104
326
y + ε/2
y
y − ε/2
x − δ/2
x
x + δ/2
subdivide
316
332
subdivide
342
340
330

Fig. 5


102
104
316
326
y + ε/2
y
y − ε/2
x − δ/2
x
x + δ/2
subdivide
316
330
332
326
y − ε/4
x + δ/4
subdivide
330
342
332
340
y − ε/8
x + δ/8

09927614.081001

Fig. 6

500 Refinement

```
500-001  refine(initial_bbox, best_estimate, best_error, depth_bound) {
500-002    subdivide initial_bbox to 4 child_bboxes
500-003    best_child_error = ∞
500-004    foreach child_bbox {
500-005      child_estimate = child_bbox.center
500-006      texture map from target to source using child_estimate
500-007      compute pixelwise child_error in source
500-008      if (child_error < best_child_error) {
500-009        best_child_error = child_error
500-010        best_child_estimate = child_estimate
500-011      }
500-012    }
500-013    if (depth_bound > 0) {
500-014      refine(child_bbox, best_child_estimate, best_child_error, depth_bound - 1)
500-015    }
500-016    if (best_child_error < best_error) {
500-017      best_error = best_child_error
500-018      best_estimate = best_child_estimate
500-019    }
500-020  }
```

Patent Application Publication

Fig. 7

450 Texture Map

```
texture_map(dx, dy, x0, y0, xf, yf) {
  glBegin(GL_QUADS);
  glTexCoord2f(x0 + dx, y0 + dy); glVertex2f(x0, y0);
  glTexCoord2f(xf + dx, y0 + dy); glVertex2f(xf, y0);
  glTexCoord2f(xf + dx, yf + dy); glVertex2f(xf, yf);
  glTexCoord2f(x0 + dx, yf + dy); glVertex2f(x0, yf);
  glEnd();
}
```